S12compact

Hardware-
Version 1.10

Benutzerhandbuch

18. März 2003

Copyright (C)2002,2003 by
MCT Elektronikladen GbR
Hohe Str. 9-13 D-04107 Leipzig
Telefon: +49-(0)341-2118354
Fax: +49-(0)341-2118355
Email: leipzig@elektronikladen.de
Web: http://www.elektronikladen.de/mct

Dieses Handbuch wurde sorgfältig erstellt und geprüft. Trotzdem können Fehler und Irrtümer nicht ausgeschlossen werden. MCT übernimmt keinerlei juristische Verantwortung für die uneingeschränkte Richtigkeit und Anwendbarkeit des Handbuchs und des beschriebenen Produktes. Die Eignung des Produktes für einen spezifischen Verwendungszweck wird nicht zugesichert. Die Haftung des Herstellers ist in jedem Fall auf den Kaufpreis des Produktes beschränkt. Eine Haftung für eventuelle Mangelfolgeschäden wird ausgeschlossen.

Produkt- und Preisänderungen bleiben, auch ohne vorherige Ankündigung, vorbehalten.

Die in diesem Handbuch erwähnten Software- und Hardwarebezeichnungen sind in den meisten Fällen auch eingetragene Warenzeichen und unterliegen als solche den gesetzlichen Bestimmungen. Es kann aus dem Fehlen einer besonderen Kennzeichnung nicht darauf geschlossen werden, daß die Bezeichnung ein freier Warenname ist. Gleiches gilt für Rechte aus Patenten und Gebrauchsmustern.

Inhalt

1. Überblick 3
Technische Daten 4
Optionale Komponenten 5
Lieferumfang 5
2. Schnellstart 6
3. Bestückungsplan 7
4. Jumper und Brücken 9
Jumper 9
Lötbrücken 9
5. Mechanische Abmessungen 11
6. Schaltungsbeschreibung 12
Schaltplan 12
Stromversorgung 12
Reseterzeugung 14
Takterzeugung und PLL 15
Betriebsarten, BDM-Unterstützung 17
Integrierter A/D-Wandler 18
Integriertes EEPROM 19
Indikator-LED 21
Akustischer Signalgeber 21
RS232-Interface 23
IF-Modul-Anschluß 24
USB-Interface 24
SPI-Subsystem 26
Zusatzports PARIN und PAROUT 27
Real Time Clock (RTC) 28

A/D-Wanndler (ADC) 30
D/A-Wandler (DAC) 31
Serial Data Flash (SDF) 32
IIC-Bus 33
CAN-Interface 33
Businterface 34
Verhalten nach Reset 34
Startup-Code 34
Zusatzinformationen im Web 35
7. Memory Map 35
Anhang 36
S-Record Format 36
EMV Hinweise 38

1. Überblick

S12compact ist eine leistungsfähige Controllerbaugruppe im kompakten Halbeuro-Format auf Basis der 16-Bit Mikrocontrollerfamilie HCS12 von Motorola. Das S12compact Modul erleichtert die Implementierung von umfangreichen Controlleranwendungen, wie z.B. Datenloggerapplikationen.

Auf dem S12compact kommt eine leistungsstarke MCU vom Typ MC9S12DG256 zum Einsatz. Dieser Mikrocontroller enthält die 16-Bit HCS12 CPU, 256KB Flash, 12KB RAM, 4KB EEPROM und eine große Menge integrierter Peripheriefunktionen, wie SCI, SPI, CAN, IIC, Timer, PWM, ADC und Input-/Output-Kanäle. Der MC9S12DG256 ist vollständig mit 16 Bit breiten internen Datenpfaden ausgestattet. Die integrierte PLL-Schaltung ermöglicht es, Performance und Strombedarf auf einfache Weise den jeweiligen Anforderungen anzupassen.

Zusätzlich zu den integrierten Controllerfunktionen bietet das S12compact Modul einige interessante Peripherieoptionen. Dazu zählen ein 16-Bit A/D-Wandler und ein 16-Bit D/A-Wandler nebst Präzisions-Spannungsreferenz, eine batteriegepufferte Echtzeituhr (RTC), ein USB-Interface und eine 2MB (16MBit) Speicheroption mit Serial Data Flash.

Die für die HCS12-Controller erhältliche umfassende Softwareunterstützung (Monitor, C-Compiler, BDM-Debugger) erleichtert die Entwicklung von Embedded Systemen jeglicher Art.

Technische Daten

- MCU MC9S12DG256 im LQFP112 Package (SMD)
- HCS12 16-Bit CPU, Programmiermodell und Befehlssatz wie beim HC12
- 16 MHz Quarztakt, bis zu 25 MHz Bustakt über PLL
- 256KB Flash
- 4KB EEPROM
- 12KB RAM
- 2x SCI - asynch. serial Interface (z.B. RS232, LIN)
- 3x SPI - synch. serial Interface
- 1x IIC - Inter-IC-Bus
- 2x msCAN-Module (CAN 2.0A/B-kompatibel)
- 8x 16-Bit Timer (Input Capture/Output Compare)
- 8x PWM (Pulse Width Modulator)
- 16-Kanal 10-Bit A/D-Wandler
- BDM - Background Debug Mode Schnittstelle
- Spezieller LVI-Schaltkreis (Reset Controller)
- BDM-Anschluß für Download und Debugging
- Serielles Interface mit RS232-Treiber, z.B. für PC-Verbindung
- Zweiter serieller Port für IF-Module (RS232, RS485, LIN...)
- Indikator-LED
- Akustischer Signalgeber
- High-Speed phys. CAN-Interface
- Resettaster
- bis zu 70 freie Ein-/Ausgabeleitungen
- acht zusätzliche digitale Eingänge
- acht zusätzliche digitale Ausgänge
- Betriebsspannung 5V, typ. Stromaufnahme ca. 70 mA
- Abmessungen 80mm x 100mm

Optionale Komponenten

- Option RTC: batteriegepufferte Echtzeituhr
- Option ADC: 8-Kanal 16 Bit A/D-Wandler (4096mV)
- Option DAC: 2-Kanal 16 Bit D/A-Wandler (4096mV)
- Option USB: Full-speed USB2.0 Interface (belegt zweiten SCI-Kanal)
- Option SDF: 2MB (16MBit) Serial Data Flash

Lieferumfang

- Controller-Baugruppe (mit Optionen gemäß Bestellung)
- D-Bug12 Monitorprogramm (im Flash Speicher der MCU)
- RS232 Anschlußkabel (Sub-D9)
- USB-Anschlußkabel (für Option /USB)
- Randsteckverbinder (zwei 72-polige Stiftleisten)
- Hardwarehandbuch (dieses Dokument)
- Schaltpläne
- CD-ROM: enthält Assemblersoftware, verschiedene Datenblätter, CPU12 Reference Manual, Softwarebeispiele, C-Compiler Demoversion u.v.m.

2. Schnellstart

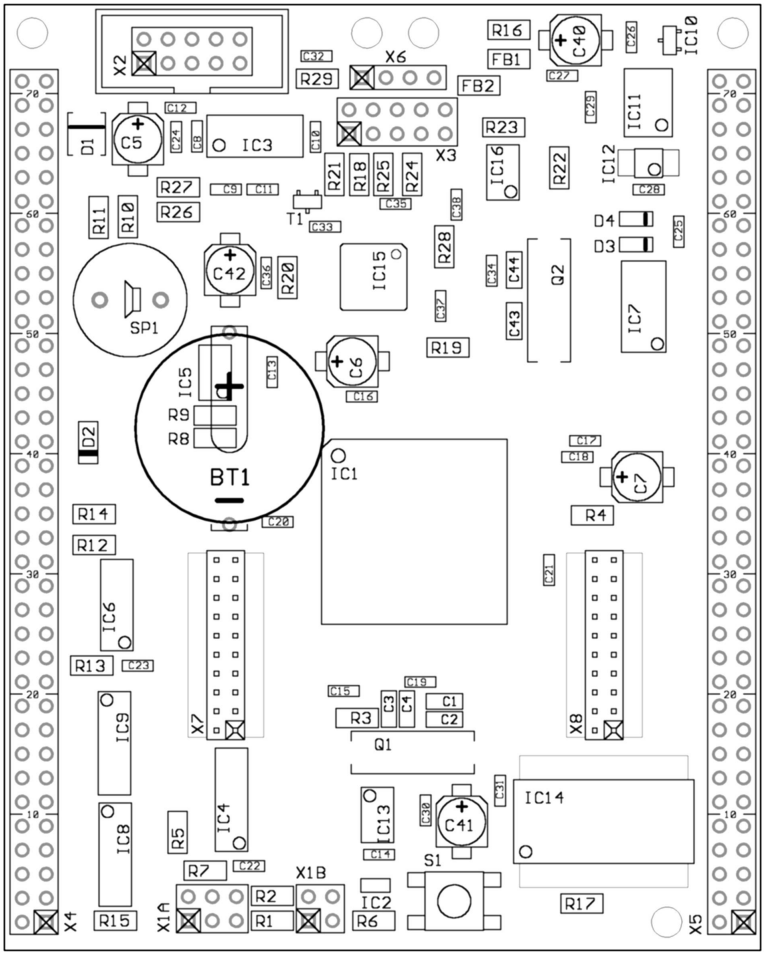
Kein Mensch liest gern dicke Handbücher. Daher hier die wichtigsten Hinweise in Kürze. Wenn Sie sich jedoch über ein Detail einmal nicht sicher sind, dann informieren Sie sich am Besten in den nachfolgenden Kapiteln.

Und so können Sie beginnen:

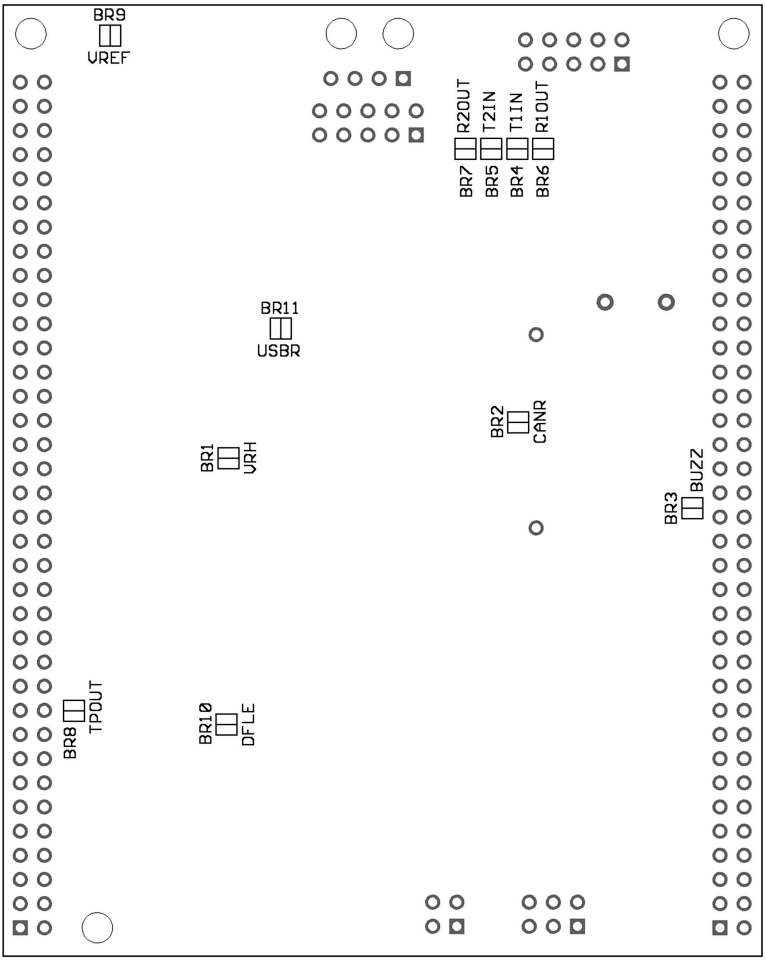
- Überprüfen Sie die Baugruppe zuerst auf offenkundige Transportschäden.
- Verbinden Sie das Controller Modul via RS232 mit Ihrem PC. Die Verbindung zwischen S12compact (Schnittstelle SCI0, Steckverbinder X2) und PC erfolgt über das mitgelieferte 10-pol. Flachbandkabel.
- Starten Sie auf dem PC ein Terminalprogramm. Ein einfaches Programm wie OC-Console (kostenlos auf unserer Website!) reicht aus.
- Stellen Sie die Baudrate auf 9600 Baud. Schalten Sie alle zusätzlichen Protokolle (Hard- und Softwarehandshake) aus.
- Schließen Sie die (stabilisierte!) Versorgungsspannung an den Einplatinenrechner an, z.B. hier:
- Masse an X4 Pin 70
- +5V an X4 Pin 72
- Vergewissern Sie sich **zuvor** von der richtigen Spannung und Polarität!
- Daraufhin startet das Monitorprogramm und zeigt eine kurze Systemmeldung an. Mit Ausgabe des Promptzeichens erwartet es Ihre Anweisungen.

Wir wünschen Ihnen viel Erfolg bei Ihrer Arbeit mit dem S12compact!

3. Bestückungsplan



Lageplan Bestückungsseite



Lötbrücken auf der Platinenrückseite

4. Jumper und Lötbrücken

Jumper

Auf dieser Baugruppe sind keine Jumper vorhanden.

Lötbrücken

Die folgenden Lötbrücken befinden sich auf der Unterseite der Platine (vergl. Lageplan auf vorhergehender Seite):

BR1: VRH

offen	externe Einspeisung VRH erforderlich
geschl.*	VRH on-board mit VDDA (VCC) verbunden

BR2: CANR

offen	keine CAN-Bus Terminierung on-board
geschl.*	CAN-Bus on-board mit 120 Ohm terminiert

BR3: BUZZ

offen	Portpin PT2 frei verfügbar
geschl.*	PT2 steuert akustischen Signalgeber (Buzzer) an

BR4: T1IN

offen	Portpin TXD0 (PS1) frei verfügbar
geschl.*	TXD0 mit RS232-Pegelwandler IC3 verbunden

BR5: T2IN

offen	Portpin PM3 frei verfügbar
geschl.*	PM3 mit RS232-Pegelwandler IC3 verbunden

BR6: R1OUT

offen	Portpin RXD0 (PS0) frei verfügbar
geschl.*	RXD0 mit RS232-Pegelwandler IC3 verbunden

* = Standardeinstellung

BR7: R2OUT

offen	Portpin PM2 frei verfügbar
geschl.*	PM2 mit RS232-Pegelwandler IC3 verbunden

BR8: TPOUT

offen*	TPOUT und /IRQ nicht verbunden
geschl.	TPOUT und /IRQ verbunden, Real Time Clock IC7 kann hierüber Interrupt auslösen

BR9: VREF

offen	externe Einspeisung VREF erforderlich
geschl.*	IC10 stellt Referenzspannung VREF bereit für 16-Bit A/D-Wandler IC11 und 16-Bit D/A-Wandler IC12

BR10: DFLE

offen	Portpin MISO1 (PH0) frei verfügbar
geschl.*	MISO1 mit Ausgangstreiber IC6B (Serial Data Flash) verbunden

BR11: USBR

offen*	Reset-Eingang USB-Transceiver IC15 und Systemreset /RESET nicht verbunden
geschl.	Systemreset /RESET löst auch ein Reset am USB-Transceiver aus

* = Standardeinstellung

5. Mechanische Abmessungen

Die folgende Tabelle gibt die mechanischen Dimensionen des S12compact Moduls wieder. Die Angaben dienen als Orientierung beim Entwurf von Trägerplatinen/-baugruppen (Achtung: Angaben stets an den gelieferten Baugruppen nachprüfen - keine Haftung für Druckfehler!).

Die "südwestliche" Ecke der Platine bildet den Koordinatenursprung. Die Lage der Platine ist horizontal, wie im Bestückungsplan (s.o.) dargestellt.

Alle Angaben zu Bohrungen (B) beziehen sich auf die Mitte, bei Steckverbindern (X) auf die Lage von Pin 1.

	X in mm	Y in mm
X1A	19,5	3,2
X1B	32,2	3,2
X2	14,8	93,8
X3	36,4	86,4
X4	4,5	3,0
X5	78,1	3,0
X6	37,8	92,3
X7	24,4	23,3
X8	64,4	23,3
B1	3,0	97,0
B2	77,0	97,0
B3	70,0	3,0
PCB	80,0	100,0

6. Schaltungsbeschreibung

Bitte beachten Sie: Dieses Hardwarehandbuch kann nur einige *spezifische* Hinweise geben. Die Behandlung *allgemeiner* Techniken zur Programmierung des Controllers in Assembler bzw. Hochsprachen würden Umfang und Ziel dieses Handbuchs sprengen. Die meisten Antworten finden Sie beim (unerläßlichen) Studium der Datenblätter und Referenzhandbüchern der Halbleiterhersteller.

Die im Text eingestreuten Beispielprogramme dienen lediglich der Demonstration. Für die Korrektheit und die Eignung für eine bestimmte Aufgabe können wir *keine Garantie* geben!

Schaltplan

Damit alle Details gut lesbar bleiben, liegt der Schaltplan im A4-Format separat bei.

Stromversorgung

Der Mikrocontroller (IC1) verfügt über drei Anschlußpaare zur Zuführung der Versorgungsspannung: VDDR/VSSR, VDDX/VSSX und VDDA/VSSA. Die Betriebsspannung beträgt nominal 5 Volt, intern arbeitet der Prozessor jedoch mit 2,5 Volt. Der hierzu erforderliche Spannungsregler ist bereits in der MCU integriert. VREGEN gibt den internen Spannungsregler frei, der Pin ist normalerweise stets mit H-Pegel (5V) zu verbinden.

Die Spannungsreduzierung im Core ist in erster Linie erforderlich durch die geringen Strukturbreiten des Fertigungsprozesses (0,25µm und kleiner). Von außen verhält sich der HCS12 jedoch wie ein 5V-Baustein, da an den Ein-/Ausgabepins Pegelwandler vorhanden sind. Eine Ausnahme stellen die Anschlüsse für Oszillator und PLL dar, näheres dazu unten.

Die drei genannten Versorgungsanschlußpaare müssen sorgfältig entkoppelt werden. In unmittelbarer Nähe der Pins befindet sich daher je ein 100nF-Keramikkondensator (C15, C16, C17), dem zusätzlich ein 10µF Elektrolytkondensator parallel geschaltet wird (C5, C6, C7).

Besonderes Augenmerk muß auf die Entkopplung des VDDA-Pfades gelegt werden, da der interne Spannungsregler aus dieser Spannung seinen Referenzwert (VDDA/2) ableitet.

Die interne 2,5-Volt-Corespannung wird an mehreren Stellen nach außen geführt, um sie dort ebenfalls entkoppeln zu können. Hierzu sind an den Anschlußpaaren VDD1/VSS1, VDD2/VSS2 sowie VDDPLL/VSSPLL weitere Keramikkapazitäten vorgesehen (C19, C20, C21). Eine statische Belastung der internen Betriebsspannung durch externe Schaltungskomponenten ist nicht statthaft! Das gilt grundsätzlich auch für VDDPLL, die als Referenzpunkt für die extern angeschlossene PLL-Filterkombination (R3, C3, C4) dient.

In die Domäne der Versorgungsspannungen fällt auch die Referenzspannung für die integrierten Analog-Digital-Wandler. Die untere Referenzspannungsgrenze wird über den Anschluß VRL festgelegt, welcher hier (wie meist üblich) auf Massepotential liegt. Die obere Referenzspannung VRH ist über die Lötbrücke BR1 mit VDDA verbunden, C18 dient hier zur Entkopplung. Um die Auflösung der internen 10-Bit A/D-Wandler voll auszuschöpfen, kann kann eine externe Referenzspannung eingespeist werden. In diesem Fall ist BR1 zu öffnen. VRH darf jedoch VDDA niemals übersteigen.

Der TEST-Pin wird nur werkseitig bei Motorola verwendet, in Anwenderschaltungen ist dieser Pin stets mit dem Massepotential zu verbinden.

Reseterzeugung

/RESET ist der bidirektionale, L-aktive Resetpin der MCU. Als Eingang dient er zur Initialisierung der MCU beim Einschalten. Als Open-Drain-Ausgang signalisiert er, dass innerhalb der MCU ein Resetereignis stattgefunden hat. Die HCS12 MCU enthält bereits Schaltungen für Power-On Reset, COP (Watchdog) and Clock Monitor Reset. Es ist dennoch notwendig, zusätzlich einen externen LVI-Schaltkreis vorzusehen, welcher die Aufgabe hat, zuverlässig Reset auszulösen, sobald die Versorgungsspannung der MCU unter den zulässigen Mindestwert gefallen ist.

Der LVI-Schaltkreis IC2 hat einen Open-Drain Ausgang, um Kollisionen mit dem bidirektionalen Resetpin der MCU zu vermeiden. Im inaktiven Zustand stellt sich an /RESET (dank des Pull-Up Widerstands R6) H-Pegel ein. Die Schaltschwelle von IC2 liegt bei typischerweise 4,6V. Das ist geringfügig höher als die Mindestbetriebsspannung der MCU (4,5V).

IC2 ist in der Lage, den Resetimpuls auf eine gewisse Mindestlänge auszudehnen, die über den Kondensator C14 festgelegt wird. Bei 100nF beträgt der Delay ca. 50..80ms.

Es ist wichtig zu bemerken, dass diese Impulsverlängerung nur bei einem Power-On Reset wirksam wird. Die MCU-internen Resetimpulse werden von IC2 hingegen nicht gedehnt, denn sonst wäre die MCU nicht mehr in der Lage, die korrekte Resetquelle zu ermitteln. Die Konsequenz wäre sonst u.U. ein Programmabsturz durch die Verwendung eines falschen Resetvektors. Es ist daher ebenso wichtig, niemals größere Kapazitäten an die Resetleitung des HCS12 anzuschliessen, denn der resultierende Effekt wäre der selbe.

Takterzeugung und PLL

Der On-Chip Oszillator des MC9S12DG256 kann den primären Takt (OSCCLK) mit Hilfe eines Quarzes (Q1) erzeugen, der an die Pins EXTAL und XTAL angeschlossen wird. Der zulässige Frequenzbereich ist 0,5 bis 16 MHz. Wie üblich sind zwei Lastkapazitäten (C1, C2) Teil der Oszillatorschaltung. Die Anordnung ist jedoch modifiziert, wenn man die Schaltung mit der Standard-Pierce Konfiguration vergleicht, wie sie beim HC11 und den meisten HC12-Typen verwendet wurde.

Der MC9S12DG256B verwendet einen Colpitts-Oszillator mit translated Ground. Der Hauptvorteil dieser Oszillatorschaltung ist eine sehr geringe Leistungsaufnahme, dafür ist die Komponentenwahl um einiges kritischer. Der S12compact verwendet einen Automotive-Quartz von NDK, mit zwei Lastkapazitäten von lediglich 3,9pF. Darüber hinaus wurde beim Design besonders auf die Minimierung von parasitären Kapazitäten geachtet, die sich nachteilig auf die Signale EXTAL und XTAL auswirken könnten.

Mit einem OSCCLK von 16 MHz ergibt sich ein Default-Bustakt (ECLK) von 8 MHz. Zur Erreichung höherer Taktfrequenzen bedient man sich der PLL-Schaltung des HCS12. Der MC9S12DG256 kann intern mit bis zu 25MHz Bustakt arbeiten, wobei die meisten Designs eine Frequenz von 24MHz nutzen, denn dies möglicht eine besonders flexible Festlegung der SCI-Baudraten.

An den Controllerpin XFC wird eine Tiefpassfilterkombination angeschlossen, sie besteht aus den Bauelementen R3, C3 und C4. Ihre Aufgabe ist die Verminderung der Welligkeit des VCO-Signals. Falls die PLL unbenutzt bleibt, kann XFC mit VDDPLL verbunden werden, andernfalls bildet VDDPLL den Bezugspotenzial für den Filter.

Die Wahl der Filterkomponenten ist stets ein Kompromiss zwischen Einschwingzeit und Stabilität der Schleife. 5 bis 10kHz Bandbreite und ein Dampingfaktor von 0,9 sind gute Startwerte für die Berechnung. Mit einer Quarzfrequenz von 16MHz und einem gewünschten Busclock von 24MHz ergibt sich eine mögliche Auswahl zu R3=4,7k und C3=22nF. C4 sollte etwa (1/20–1/10) x C3 betragen, hier also 2,2nF. Das Kapitel "XFC Component Selection” im

MC9S12DP256B Device User Guide illustriert die erforderlichen Rechenschritte.

Das folgende Listing zeigt die erforderlichen Initialisierungsschritte für die PLL:

```
//=============================================================================
// File: S12_CRG.C - V1.00
//=============================================================================

//-- Includes ---------------------------------------------------------------

#include <hcs12dp256.h>
#include "s12_crg.h"

//-- Code -------------------------------------------------------------------

void initPLL(void) {

    CLKSEL &= ~BM_PLLSEL;               // make sure PLL is *not* in use
    PLLCTL |= BM_PLLON+BM_AUTO;         // enable PLL module, Auto Mode
    REFDV = S12_REFDV;                  // set up Reference Divider
    SYNR = S12_SYNR;                    // set up Synthesizer Multiplier
    // the following dummy write has no effect except consuming some cycles,
    // this is a workaround for erratum MUCTS00174 (mask set 0K36N only)
    // CRGFLG = 0;
    while((CRGFLG & BM_LOCK) == 0) ;    // wait until PLL is locked
    CLKSEL |= BM_PLLSEL;                // switch over to PLL clock
    }

//=============================================================================
```

Alternativ zur Takterzeugung mit Q1 kann über den EXTAL-Pin des MC9S12DG256B ein externer Takt eingespeist werden. /XCLKS muss hierzu während Reset auf L-Pegel gelegt werden. Da von dieser Variante auf dem S12compact Modul keine Verwendung gemacht wird, dient R5 dazu, /XCLKS während Reset auf H-Pegel zu halten. Achtung: andere HCS12-Typen haben z.T. abweichende Funktionalität hinsichtlich des /XCLKS-Pins.

Betriebsarten, BDM-Unterstützung

Drei Pins des HCS12 dienen der Auswahl der MCU-Betriebsart: MODA, MODB und BKGD (=MODC). MODA und MODB werden durch die Widerstände R1 und R2 auf L-Pegel gebracht, um Single Chip Mode auszuwählen. BKGD ist über R7 mit H-Pegel verbunden, damit die MCU im Normal Single Chip Mode startet. Dies ist die übliche Betriebsart zur Abarbeitung von Anwendungsprogrammen.

Die HCS12 Betriebsart, welche für Download und Debugging genutzt wird, heisst Background Debug Mode (BDM). BDM ist direkt nach Reset aktiv, wenn die MCU im Special Single Chip Mode betrieben wird. Dies wird erreicht, indem - zusätzlich zu MODA und MODB - auch die BKGD-Leitung während Reset vorübergehend auf L-Pegel gebracht wird.

Zwischen beiden Modi kann man leicht umschalten, da sich lediglich der Resetzustand der BKGD-Leitung unterscheidet. Ein BDM-Pod, welches am Steckverbinder X1A angeschlossen wird, kann die Umschaltung automatisch vornehmen, und macht einen mechanischen Umschalter überflüssig. Das BDM-Pod wäre ohnehin notwendig zum BDM-basierten Download von Software bzw. als Debugger, gesteuert von Software auf einem (Entwicklungs-) PC.

Der 6-pol. Steckverbinder X1A folgt der Motorola-Standardbelegung für BDM12-Anschlüsse. Steckverbinder X1B trägt einige zusätzliche Signale, welche normalerweise für das BDM-Debugging nicht zwingend erforderlich sind. Einige Debugger jedoch bieten zusätzliche Features, welche das Vorhandensein dieser Signale voraussetzen.

Integrierter A/D-Wandler

Der MC9S12DG256 verfügt über zwei integrierte Analog/Digital-Wandler Module mit einer Auflösung von max. 10 Bit. Beide Module (ATD0, ATD1) haben jeweils acht gemultiplexte Eingänge.

Die Referenzspannung VRH legt die obere Grenze der Eingangsspannung aller A/D-Kanäle fest, sie ist auf dem S12compact ab Werk über BR1 mit VDDA (5V) verbunden. Durch Öffnen der Lötbrücke BR1 ist es möglich, über X5/46 eine externe Referenzspannung einzuspeisen. Denkbar wäre z.B., auf die Referenzspannungspannungsquelle IC10 zurückzugreifen, die eine Spannung von 4,096V an X5/71 bereitstellt (vorausgesetzt, IC10 ist bestückt).

Das folgende Beispielprogramm zeigt die Initialisierungssequenz für das A/D-Wandler Modul ATD0 und eine Routine zum Erfassen des Spannungswertes eines einzelnen Eingangskanals. Weitere Beispielroutinen für das integrierte ATD-Modul sind in der Quelltextdatei S12_ATD.C enthalten.

```
//============================================================================
// File: S12_ATD.C - V1.00
//============================================================================

//-- Includes ----------------------------------------------------------------

#include "datatypes.h"
#include <hcs12dp256.h>
#include "s12_atd.h"

//-- Code --------------------------------------------------------------------

// Func: Initialize ATD module
// Args: -
// Retn: -
//
void initATD0(void) {

    // enable ATD module
    ATD0CTL2 = BM_ADPU;
    // 10 bit resolution, clock divider=12 (allows ECLK=6..24MHz)
    // 2nd sample time = 2 ATD clocks
    ATD0CTL4 = BM_PRS2 | BM_PRS0;
    }

//----------------------------------------------------------------------------

// Func: Perform single channel ATD conversion
// Args: channel = 0..7
// Retn: unsigned, left justified 10 bit result
//
UINT16 getATD0(UINT8 channel) {

    // select one conversion per sequence
    ATD0CTL3 = BM_S1C;
    // right justified unsigned data mode
    // perform single sequence, one out of 8 channels
    ATD0CTL5 = BM_DJM | (channel & 0x07);
```

```
    // wait until Sequence Complete Flag set
    // CAUTION: no loop time limit implemented!
    while((ATD0STAT0 & BM_SCF) == 0) ;
    // read result register
    return ATD0DR0;
    }

//-----------------------------------------------------------------------------
```

Integriertes EEPROM

Der interne EEPROM-Speicher des MC9S12DG256 ist 4KB groß und in 1024 Sektoren zu je 4 Byte (32 Bit) unterteilt. Gelöscht wird stets sektorweise (4 Byte), während die Programmierung wortweise (2 Byte) erfolgen kann. Lesezugriffe auf den EEPROM erfolgen beliebig, also byte- oder wortweise.

Nach Reset ist der EEPROM Bereich im MC9S12DG256 ab Adresse 0 gemappt, wird dadurch aber partiell (0x0000..0x03FF) von den Steuerregistern überlagert. Will man nicht auf die ersten 1024 Byte des EEPROM verzichten, muss der EEPROM-Bereich verschoben werden (INITEE Register).

Das folgende Beispiel beläßt den EEPROM auf der Defaultposition, in der Initialisierungsroutine wird lediglich der EEPROM Clock Devider entsprechend der Quarzfrequenz des S12compact eingestellt. Die Schreibfunktion wrSectEETS() kopiert zwei Worte (4 Byte) von einer beliebigen Quelladresse src auf eine EEPROM-Adresse dest, letztere muß identisch mit einer EEPROM-Sektorgrenze sein (aligned 32 bit). Ist der Inhalt des Zielsektors nicht gelöscht (0xFFFFFFFF), wird zunächst automatisch ein Sector-Erase ausgeführt.

Die Zugriffsfunktionen readItemEETS() und writeItemEETS() verallgemeinern den EEPROM-Zugriff dahin gehend, dass nicht mehr mit EEPROM-Adressen gearbeitet wird, sondern mit einer abstrakten Numerierung von EEPROM-"Items". Jedes dieser EEPROM-"Items" kann 1 bis 4 Byte lang sein.

```
//=============================================================================
// File: S12_EETS.C - V1.00
//=============================================================================

//-- Includes ------------------------------------------------------------------

#include "datatypes.h"
#include <hcs12dp256.h>
#include "s12_eets.h"

//-- Code ----------------------------------------------------------------------

void initEETS(void) {

    ECLKDIV = EETS_ECLKDIV;         // set EEPROM Clock Divider Register
    }

//-----------------------------------------------------------------------------

INT8 wrSectEETS(UINT16 *dest, UINT16 *src) {

    // check addr: must be aligned 32 bit
    if((UINT16)dest & 0x0003) return -1;
    // check if ECLKDIV was written
    if((ECLKDIV & BM_EDIVLD) == 0) return -2;
    // make sure error flags are reset
    ESTAT = BM_PVIOL | BM_ACCERR;
    // check if command buffer is ready
    if((ESTAT & BM_CBEIF) == 0) return -3;
    // check if sector is erased
    if((*dest != 0xffff) || (*(dest+1) != 0xffff)) {
        // no, go erase sector
        *dest = *src;
        ECMD = EETS_CMD_SERASE;
        ESTAT = BM_CBEIF;
        if(ESTAT & (BM_PVIOL | BM_ACCERR)) return -4;
        while((ESTAT & BM_CBEIF) == 0) ;
        }
    // program 1st word
    *dest = *src;
    ECMD = EETS_CMD_PROGRAM;
    ESTAT = BM_CBEIF;
    if(ESTAT & (BM_PVIOL | BM_ACCERR)) return -5;
    while((ESTAT & BM_CBEIF) == 0) ;
    // program 2nd word
    *(dest+1) = *(src+1);
    ECMD = EETS_CMD_PROGRAM;
    ESTAT = BM_CBEIF;
    if(ESTAT & (BM_PVIOL | BM_ACCERR)) return -6;
    while((ESTAT & BM_CCIF) == 0) ;
    return 0;
    }

//-----------------------------------------------------------------------------

INT8 writeItemEETS(UINT16 item_no, void *item) {

    if(item_no >= EETS_MAX_SECTOR) return -7;
    item_no = EETS_START + (item_no << 2);
    return wrSectEETS((UINT16 *)item_no, (UINT16 *)item);
    }

//-----------------------------------------------------------------------------

INT8 readItemEETS(UINT16 item_no, void *item) {

    if(item_no >= EETS_MAX_SECTOR) return -7;
    item_no = EETS_START + (item_no << 2);
    *((UINT16 *)item) = *((UINT16 *)item_no);
    *(((UINT16 *)item)+1) = *(((UINT16 *)item_no)+1);
    return 0;
    }

//=============================================================================
```

Indikator-LED

Am Portpin PE6 dient der Pegel des Steuersignals /XCLKS zur Auswahl der Clock-Konfiguration des MC9S12DG256. Führt das Signal H-Pegel, wird der integrierte Colpitts-Oszillator aktiviert. Relevant ist dabei der Zustand zum Zeitpunkt der steigenden Resetflanke. Danach steht PE6 als General-Purpose-I/O zur Verfügung. Auf dem S12compact dient dieses Signal dann zur Ansteuerung der Indikator-LED D2, getrieben vom Buffer IC6C.

Der Steuerung der Indikator-LED kann durch einige einfache Makros erfolgen, wie das folgende Headerfile zeigt.

```
//============================================================================
// File: S12CO_LED.H - V1.00
//============================================================================

#ifndef __S12CO_LED_H
#define __S12CO_LED_H

//-- Macros ------------------------------------------------------------------

#define initLED()   PORTE |= 0x80; DDRE |= 0x80
#define offLED()    PORTE |= 0x80
#define onLED()     PORTE &= ~0x80
#define toggleLED() PORTE ^= 0x80

//-- Function Prototypes -----------------------------------------------------

/* module contains no code */

#endif //__S12CO_LED_H ===================================================
```

Akustischer Signalgeber

Der Signalgeber SP1 wird durch den Buffer IC6D getrieben und wird von der MCU über den Portpin PT2 angesteuert, es sei denn, die Lötbrücke BR3 ist offen.

PT2 ist mit einem der acht Timerkanäle der MCU verbunden. Dies ermöglicht die Frequenzerzeugung mit Hilfe der Output-Compare Funktion des Timersystems.

Das folgende Beispielmodul demonstriert anhand weniger Zeilen, wie eine interruptgesteuerte Output-Compare Funktion eingesetzt werden kann, um Tonfrequenzen zu erzeugen:

```
//=============================================================================
// File: ACPRD_FREQOUT.C - V1.00
//=============================================================================

//-- Includes ---------------------------------------------------------------

#include "datatypes.h"
#include "hcs12dp256.h"
#include "s12_ect.h"
#include "s12_crg.h"                    // contains S12_ECLK value
#include "acprd_freqout.h"

//-- Static Vars ------------------------------------------------------------

UINT16 freqout_tticks;

//-- Code -------------------------------------------------------------------

void initFreqOut(void) {

    // make sure timer is enabled
    TSCR1  |= BM_TEN;
    // prescaler = 2**4 = 16
    TSCR2 = 0x04;
    // select Output Compare function for channel 2
    TIOS |= BM_2;
    DDRT |= BM_2;
    // enable Interrupt for channel 2
    TIE |= BM_2;
    // timer disconnected from PT2 pin
    TCTL2 &= ~(BM_OM2 | BM_OL2);
    }

//-----------------------------------------------------------------------------

// period is in µs
//
void setFreqOut(UINT16 period) {

    UINT16 tticks;

    tticks = period * (S12_ECLK / 2000000L);
    tticks /= TIMER_TCNT_PRE;

    if(period == 0) {
        // disconnect PT2 pin
        TCTL2 &= ~(BM_OM2 | BM_OL2);
        }
    else {
        // connect PT2 pin
        TCTL2 |= BM_OL2;
        }
    freqout_tticks = tticks;
    }

//-----------------------------------------------------------------------------

// OC2 toggles buzzer
//
#ifdef METROWERKS_C
interrupt
#endif
#ifdef IMAGECRAFT_C
#pragma interrupt_handler isrOC2
#endif
void isrOC2(void) {

    TC2 += freqout_tticks;
    TFLG1 = BM_2;                       // clear Intr flag
    }

//=============================================================================
```

RS232-Interface

Der MC9S12DG256 verfügt über zwei asynchrone Schnittstellen (SCI0, SCI1). Jede dieser Schnittstellen umfaßt zwei Signalleitungen (RXDx, TXDx). Handshakeleitungen sind nicht Bestandteil der SCI-Module des Controllers, sie sind durch Einbeziehung zusätzlicher I/O-Ports zu realisieren.

SCI0 dient auf dem S12compact als RS232-Interface. Für diese Schnittstelle können auf dem Modul die Ports PM2 und PM3 als Handshakesignale eingesetzt werden. Diese Portsignale lassen sich über die Lötbrücken BR7 und BR5 mit dem RS232-Pegelwandler IC3 verbinden. Ebenso sind die Signalleitungen RXD0 und TXD0 über Brücken (BR6, BR4) mit IC3 verbunden. Öffnet man diese Lötbrücken, können die Controllersignale anderweitig verwendet werden. Sie werden dazu am Steckverbinder X5 bereitgestellt.

Der RS232-Anschluß erfolgt über X2. Dieser Steckverbinder ist so gestaltet, dass durch ein Flachbandkabel mit angecrimpter Sub-D9 Buchse eine direkte Verbindung zu einem PC-COM-Port hergestellt werden kann.

Das Codebeispiel zeigt die Ansteuerung von SCI0 mittels Polling (Achtung: für Interrupt-Betrieb beachten Sie bitte die Errata-Information zum MC9S12DG256B von Motorola!):

```
//============================================================================
// File: S12_SCI.C - V1.00
//============================================================================

//-- Includes ----------------------------------------------------------------

#include "datatypes.h"
#include <hcs12dp256.h>
#include "s12_sci.h"

//-- Code --------------------------------------------------------------------

void initSCI0(UINT16 bauddiv) {

    SCI0BD  = bauddiv & 0x1fff; // baudrate divider has 13 bits
    SCI0CR1 = 0;                // mode = 8N1
    SCI0CR2 = BM_TE+BM_RE;      // Transmitter + Receiver enable
    }

//----------------------------------------------------------------------------

UINT8 getSCI0(void) {

    while((SCI0SR1 & BM_RDRF) == 0) ;
    return SCI0DRL;
    }
```

```
//---------------------------------------------------------------------------

void putSCI0(UINT8 c) {

    while((SCI0SR1 & BM_TDRE) == 0) ;
    SCI0DRL = c;
    }

//---------------------------------------------------------------------------
```

IF-Modul-Anschluß

SCI1 kann auf dem S12compact auf zwei Arten Verwendung finden. Falls die USB-Option bestückt ist, dient SCI1 zur Ansteuerung des USB-Transceivers IC15 (weitere Informationen zur USB-Option s.u.). Andernfalls steht die Schnittstelle als universelle TTL-RS232 (ohne Pegelwandler) zur Verfügung, um externe IF-Module anzusteuern.

IF-Module sind anschlußkompatible, serielle Interfacemodule, die für verschiedene physikalische Schnittstellenvarianten zur Verfügung stehen, z.B. RS232, RS485, Strombauchinterface oder LIN. Der Anschluß von IF-Modulen an den S12compact (X3) erfolgt über ein 10-poliges Flachbandkabel.

Dem SCI1-Modul sind als Handshakeleitungen die Controllersignale PM4..7 zugeordnet. Wird an X3 kein IF-Modul betrieben, und ist auch die USB-Option nicht bestückt, können diese Signale (inkl. RXD1 und TXD1) für andere Zwecke verwendet werden. Sie stehen am Steckverbinder X4 bzw. X5 zur Verfügung.

USB-Interface

IC15 ist ein USB-UART vom Typ FT232BM. Er realisiert die bidirektionale, transparente Umsetzung einer asynchronen seriellen Schnittstelle in das USB-Protokoll, gemäß der derzeit gültigen USB-Spezifikation 2.0. Der FT232BM gehört der Klasse der USB "Full-Speed" Devices an.

Die Daten werden vom HCS12 via SCI1 an den USB-UART übermittelt, dort auf USB umgesetzt und schließlich im PC über einen virtuellen COM-Port bereit gestellt. Die hierzu erforderliche Treibersoftware stellt der Hersteller FTDI für die Anwender des FT232BM abgabenfrei zur Verfügung (siehe http://www.ftdichip.com). Zum

Manuskriptzeitpunkt waren Treiber für Windows-PCs (98 bis XP), Apple-Rechner und Linuxsysteme verfügbar. Auch auf der S12compact Produkt-CD sind die Treiber (Windows-Plattform) enthalten, ggf. existieren aber bereits neuere Versionen.

Zwecks Datentransfer zwischen USB-UART und Mikrocontroller sind ausschließlich die Signale TXD und RXD (TXD1 und RXD1 an der MCU) erforderlich. Bei Bedarf kann zusätzlich ein Hardwareprotokoll über /RTS und /CTS implementiert werden. Den /RTS-Ausgang wertet die MCU über ihren Portpin PM6 aus und steuert mit PM7 den /CTS-Eingang des USB-UARTs an.

Sende- bzw. Empfangsaktivitäten signalisiert der USB-UART bei Bedarf über zwei Leuchtdioden (LEDs). Die Anoden dieser LEDs werden mit VCC (5V) verbunden, die Kathoden an /RXLED bzw. /TXLED angeschlossen (X4/67+68).

Über den Anschluß /PWREN zeigt IC15 an, ob die USB-Enumerierung abgeschlossen ist. Dies kann über das Controllersignal PM4 ausgewertet werden. Das Kontrollsignal /SLEEP zeigt an, dass sich IC15 im Suspend-Mode befindet. Über Controllerpin PM5 kann der Zustand dieses Signals ausgewertet werden.

Soll durch ein Systemreset auch der USB-UART zurückgesetzt werden, ist BR11 zu schließen. Ein Systemreset bedeutet in diesem Fall stets eine zeitweise Abkopplung des USB-Device vom Bus mit nachfolgender Re-Enumeration. Im Defaultzustand ist BR11 offen, dennoch ist ein korrektes Power-On Reset des USB-UART gewährleistet.

Der serielle EEPROM IC16 kann Konfigurationsdaten für den USB-UART enthalten, er ist jedoch im Lieferzustand gelöscht. Der USB-UART antwortet dann auf einen Descriptor-Request mit seinen Standarddescriptoren. User Descriptoren (VID, PID, Strings, Seriennummern etc.) können im EEPROM mit Hilfe eines PC-basierten Utility-Programms abgelegt werden. Die Programmierung erfolgt in-circuit via USB.

Hinweis: Ist die USB-Option bestückt, steht der IF-Modul Anschluß X3 nicht mehr zur Verfügung.

SPI-Subsystem

Der MC9S12DG256 verfügt über drei unabhängige SPI-Ports. Auf dem S12compact wird der Port SPI0 verwendet, um die Peripheriekomponenten RTC, ADC, DAC, PARIN und PAROUT seriell anzusteuern. Am Port SPI1 ist die Peripherieoption Serial Data Flash angeschlossen; SPI2 ist unbenutzt.

SPI0 umfasst die Leitungen MISO, MOSI, SCK und /SS, das sind die MCU-Portleitungen PS4 bis PS7. /SS wird in der Schaltung des S12compact selbst nicht benutzt, dieses Signal wird lediglich an die seitlichen Stiftleisten geführt.

Die Chipselectsignale werden von den MCU-Portleitungen PH4..PH6 abgeleitet. Der 1-aus-8-Dekoder IC4 aktiviert stets einen Ausgang, abhängig vom Bitmuster, welches von Port H stammt. Dies ist ein ökonomischer Weg, um mit wenigen MCU-Leitungen bis zu acht Chipselectsignale zu betätigen. Die folgende Tabelle zeigt, wie die einzelnen Chipselectsignale zugeordnet sind:

Chipselect	Port H	Verwendung
/SPICS0	x000xxxx	Real Time Clock
/SPICS1	x001xxxx	D/A-Wandler
/SPICS2	x010xxxx	A/D-Wandler
/SPICS3	x011xxxx	Parallel In Shift
/SPICS4	x100xxxx	Parallel In Latch
/SPICS5	x101xxxx	Parallel Out
/SPICS6	x110xxxx	frei verfügbar an X4/20
/SPICS7	x111xxxx	inaktiv

Das folgende Listing zeigt die Basisfunktionen (Initialisierung, 8-Bit Datentransfer) für den SPI-Port SPI0:

```
//============================================================================
// File: S12_SPI.C - V1.00
//============================================================================

//-- Includes ------------------------------------------------------------

#include "datatypes.h"
#include <hcs12dp256.h>
#include "s12_spi.h"

//-- Code ----------------------------------------------------------------

void initSPI0(UINT8 bauddiv, UINT8 cpol, UINT8 cpha) {

    DDRS |= 0xe0;                       // SS,SCK,MOSI Output
    SPI0BR = bauddiv;                   // set SPI Rate
    // enable SPI, Master Mode, select clock polarity/phase
    SPI0CR1 = BM_SPE | BM_MSTR | (cpol ? BM_CPOL : 0) | (cpha ? BM_CPHA : 0);
    SPI0CR2 = 0;                        // as default
    }

//----------------------------------------------------------------------------

UINT8 xferSPI0(UINT8 abyte) {

    SPI0DR = abyte;                     // start transfer
    while((SPI0SR & BM_SPIF) == 0) ;    // wait until transfer finished
    return(SPI0DR);                     // read back data received
    }

//============================================================================
```

Zusatzports PARIN und PAROUT

Der MC9S12DG256 bietet eine Vielzahl universeller Ein-/Ausgabeleitungen. In manchen Applikationen ist es jedoch wünschenswert, zusätzliche Anschlußmöglichkeiten zu schaffen.

Zusätzliche Ein-/Ausgabeports können vorzugsweise über eines der Serial Peripheral Interface (SPI) Systeme des Controllers realisiert werden. Wie einfach es ist, weitere acht Ausgabeleitungen zu schaffen, zeigt das Schieberegister IC9. Zusätzlich zu den SPI-Signalen MOSI und SCK ist lediglich ein Chipselectsignal (/SPICS5) erforderlich, um IC9 anzusteuern. Das Schieberegister hat einen asynchronen Reseteingang, sowohl für das Schieberegister als auch für das integrierte Ausgabelatch. Daher führen alle Ausgänge (PO0..PO7) sofort nach Reset L-Pegel.

Das folgende Listing zeigt ein Beispiel für die Parallel-Ausgabe:

```
//-----------------------------------------------------------------------------

void putSPIPO(UINT8 abyte) {

    // set up SPI mode
    SPI0CR1 = BM_SPE | BM_MSTR;         // CPOL=0 CPHA=0
    // send data
    PTH = S12CO_SPICS5;                 // enable /CS5
    xferSPI0(abyte);
    PTH = S12CO_SPICSH;                 // disable /CS5
    }

//-----------------------------------------------------------------------------
```

Am selben SPI-Anschluß sorgt IC8 für acht zusätzliche Eingabeleitungen. Zunächst ist Chipselectleitung /SPICS4 zu betätigen, um die Eingangssignale an PI0..PI7 zu latchen. Danach wird Chipselectleitung /SPICS3 aktiviert, um die Information via MOSI seriell auslesen zu können. IC6A wird benötigt, um den Push-Pull-Ausgang QH des IC8 von MISO abzukoppeln, damit die Kooperation aller SPI-Slaves am selben SPI-Port gesichert ist.

Das folgende Beispiel illustriert die Parallel-Eingabe Funktion:

```
//-----------------------------------------------------------------------------

UINT8 getSPIPI(void) {

    UINT8 abyte;

    // set up SPI mode
    SPI0CR1 = BM_SPE | BM_MSTR;         // CPOL=0 CPHA=0
    // latch input data
    PTH = S12CO_SPICS4;
    PTH = S12CO_SPICSH;
    // serialize latched data
    PTH = S12CO_SPICS3;
    abyte = xferSPI0(0);
    PTH = S12CO_SPICSH;
    return abyte;
    }

//-----------------------------------------------------------------------------
```

Real Time Clock (RTC)

Auf dem S12compact befindet sich - als Bestückungsoption - eine Real Time Clock (Echtzeituhr, RTC) vom Typ RTC4553. Diese RTC stellt eine Zeitreferenz inkl. Kalenderinformation bereit. Sie wird bei Stromausfall bzw. -abschaltung von einer Lithium-Zelle (BT1) gepuffert. Der installierte Typ ist auf eine Lebensdauer von mindestens 5 Jahren konzipiert.

Die RTC wird über die SPI0-Schnittstelle des HCS12 angesprochen. Die benötigten SPI-Signale (MISO, MOSI, SCK) liegen auf Leitungen des Port S (PS4/5/6) der MCU. Das Chipselectsignal /SPICS0 wird von IC4 bereit gestellt (s.o.). Die Schreib-/Lesesteuerung erfolgt über PH7.

Im folgenden Listing sind grundlegende Ein-/Ausgabe-Routinen für die RTC gezeigt, sie bauen auf den zuvor gezeigten SPI Basisroutinen auf. Auf die Darstellung komplexere Zugriffsfunktionen, wie z.B. das Lesen oder Setzen von Datum und Uhrzeit, wird hier verzichtet. Entsprechende Softwarebeispiele befinden sich auf der Produkt-CD. Details zur Programmierung der RTC können dem RTC4553 Application Manual entnommen werden.

```
//-----------------------------------------------------------------------------

// Func: getRTC()
// Args: regno[0..3] is RTC register number
// Retn: [0..3] RTC data (register contents)
// Note: there is no need to range check the regno argument since the
//       RTC will ignore any additional bits anyway
//
UINT8 getRTC(UINT8 regno) {

    UINT8 result;

    // set up SPI mode: enable, master, CPOL=1, CPHA=1, LSB first
    SPI0CR1 = BM_SPE | BM_MSTR | BM_CPOL | BM_CPHA | BM_LSBFE;
    // transfer data
    PTH = S12CO_SPICS0_RD;              // enable /CS0 (reading)
    xferSPI0(regno);                    // send register number to RTC
    result = xferSPI0(0) >> 4;          // receive data from RTC
    PTH = S12CO_SPICSH;                 // disable all /CSx
    return result;
    }

//-----------------------------------------------------------------------------

// Func: putRTC()
// Args: regno[0..3] is RTC register number
//       data[0..3] is RTC data (for that register)
// Retn: -
//
void putRTC(UINT8 regno, UINT8 data) {

    // set up SPI mode: enable, master, CPOL=1, CPHA=1, LSB first
    SPI0CR1 = BM_SPE | BM_MSTR | BM_CPOL | BM_CPHA | BM_LSBFE;
    // transfer data
    PTH = S12CO_SPICS0_WR;              // enable /CS0 (writing)
    xferSPI0((data << 4) | (regno & 0x0f));
    PTH = S12CO_SPICSH;                 // disable all /CSx
    }

//-----------------------------------------------------------------------------
```

A/D-Wandler (ADC)

Der S12compact verfügt als Bestückungsoption über einen hochauflösenden A/D-Wandler (ADC), welcher zusätzlich zu den integrierten 10-Bit ATD-Kanälen der MCU die Möglichkeit bietet, Analogwerte mit hoher Genauigkeit zu erfassen.

Der ADS8344 (IC11) von Burr Brown ist ein 8-Kanal 16-Bit ADC mit seriellem Interface und einer Umsetzrate von bis zu 100ksps. Er ist mit dem SPI0-Modul der MCU verbunden und wird durch das von IC4 erzeugte Chipselectsignal /SPICS2 ausgewählt.

Die Referenzspannung wird von IC10 erzeugt, sie beträgt 4,096V. Diese ist zugleich die obere Eingangsspannungsgrenze an den Eingängen AIN0..AIN7. Die untere Grenze ist das Massepotential. Bei Bedarf kann statt der On-Board Referenz eine externe Referenzspannung eingespeist werden. Hierzu ist die Lötbrücke BR9 zu öffnen (Achtung: Auswirkung auch auf den D/A-Wandler IC12). Die Eingangssignale AIN0 bis AIN7 sowie die Versorgungsspannungen des Analogteils (VCCA, VREF, GNDA) sind am Steckverbinder X4 verfügbar.

Das folgende Beispiel zeigt, wie der ADS8344, aufbauend auf den oben geschilderten SPI-Basisfunktionen, angesteuert werden kann.

```
//-------------------------------------------------------------------------------
// Note: CPHA=0, CPOL=0 required
//       clock rate max. 2MHz
//       conversion takes max. 8µs
//
UINT16 getADS8344(UINT8 channel) {

    volatile UINT8 n;
    UINT16 adcresult;

    // set up SPI mode
    SPI0CR1 = BM_SPE | BM_MSTR;         // CPOL=0 CPHA=0
    // send conversion command
    PTH = S12CO_SPICS2;                 // enable /CS2
    xferSPI0((channel << 4) | 0x86);    // single ended, internal clock mode
    PTH = S12CO_SPICSH;                 // disable /CS2
    // wait 8µs
    for(n=0; n<100; n++) ;
    // get conversion result
    PTH = S12CO_SPICS2;                 // enable /CS2
    adcresult = xferSPI0(0) << 8;       // get bits 15..9
    adcresult += xferSPI0(0);           // get bits 8..1
    adcresult <<= 1;
    if(xferSPI0(0) & 0x80) adcresult++; // get bit 0
    PTH = S12CO_SPICSH;                 // disable /CS2
    return adcresult;
    }

//-------------------------------------------------------------------------------
```

D/A-Wandler (DAC)

Der 16-Bit D/A-Wandler IC12 ist einen weitere Schaltungsoption des S12compact. Zum Einsatz kommt ein DAC8532 von Burr Brown. Dieser Baustein bietet zwei Kanäle, deren Ausgangsspannungen VOUTA und VOUTB sowohl nacheinander als auch gleichzeitig aktualisiert werden können.

IC10 liefert die Referenzspannung (4,096V), welche die obere Ausgangsspannungsgrenze vorgibt. Sie kann durch eine externe Referenzspannungsquelle ersetzt werden kann, wenn zuvor BR9 aufgetrennt wird (Auswirkung auch auf den ADC!).

Der Lastwiderstand an den Ausgängen sollte 2kOhm nicht unterschreiten. Beim Einschalten der Betriebsspannung wird die Ausgangsspannung der beiden DAC-Kanäle im IC auf 0V zurückgesetzt.

Der DAC wird - wie auch RTC, ADC und die Zusatz-I/Os - über SPI0 angesteuert. /SPICS1 dient das Chipselectsignal für den DAC.

Die Ausgangssignale VOUTA und VOUTB sind am Steckverbinder X4 verfügbar.

Die folgende C-Funktion zeigt ein Beispiel zur Ansteuerung der D/A-Kanäle:

```
//------------------------------------------------------------------------------

// Note: CPHA=0, CPOL=0 required
//
void putDAC8532(UINT8 channel, UINT16 value) {

    // set up SPI mode
    SPI0CR1 = BM_SPE | BM_MSTR | BM_CPHA; // CPOL=0 CPHA=1
    // send command
    PTH = S12CO_SPICS1;                 // enable /CS1
    if((channel & 1) == 0)
        xferSPI0(0x10);                 // load & set DACA
    else
        xferSPI0(0x24);                 // load & set DACB
    // send data
    xferSPI0(value >> 8);               // send bits 15..8
    xferSPI0(value);                    // send bits 7..0
    PTH = S12CO_SPICSH;                 // disable /CS1
    }

//------------------------------------------------------------------------------
```

Serial Data Flash (SDF)

Die Schaltungsoption Serial Data Flash (SDF) des S12compact ermöglicht es, sequentiell anfallende Daten mit hoher Geschwindigkeit abzuspeichern, wie es z.B. bei Datenloggerapplikationen erforderlich ist. Der Nachteil des Einsatzes von Flash als nichtflüchtiger Speicher besteht normalerweise darin, dass ein relativ umfangreicher RAM-Bereich bereit gestellt werden muss, um die anfallenden Daten zwischenzupuffern.

Diesen Nachteil hat Atmel bei den Serial Data Flash Bausteinen beseitigt. Zwei integrierte RAM-Puffer mit je 528 Byte ermöglichen es, Datenübertragung und Schreibvorgang zu parallelisieren. Während ein Puffer via SPI gefüllt wird, wird der Inhalt des anderen Puffers in den Flash programmiert, welcher zu diesem Zweck ebenfalls in 528 Byte große Pages unterteilt ist. Somit wird ein kontinuierlicher Datenstrom ermöglicht (ca. 35KB/s schreibend).

Der Serial Data Flash (IC14) wird über das SPI1-Modul der MCU betrieben und ist somit unabhängig von RTC, ADC, DAC usw., welche über das SPI0-Modul angesprochen werden. Die Datensignale MISO1, MOSI1, das Taktsignal SCK1 und das Chipselectsignal /SS1 werden von der MCU über die Portpins PH0..PH3 abgewickelt.

Hinweis: Um diese Belegung zu erreichen, muß Bit 5 im Module Routing Register (MODRR) des MC9S12DG256 gesetzt werden, andernfalls (Resetzustand) ist SPI1 mit den Portpins PP0..PP3 verknüpft.

Das Flasharray benötigt keine zusätzliche Programmierspannung, die erforderlichen Timings erzeugt der Baustein ebenfalls intern. Die Versorgungsspannung für IC14 beträgt 3,3V, sie wird von einem Low-Dropout Linearregler (IC13) bereitgestellt. Die Eingänge des Serial Data Flash Bausteins sind 5V-tolerant, der Ausgang SO wird über einen Levelshifter (IC6B) an das 5V-basierte MISO1-Signal angepasst.

Falls IC14 nicht bestückt wird, erhält IC6B über R17 dennoch einen definierten Eingangspegel. MISO1 kann durch Öffnen der Lötbrücke BR10 vom Ausgang des IC6B entkoppelt werden.

IIC-Bus

An den Pins PJ6 und PJ7 bietet der MC9S12DG256 bei Bedarf einen Inter-IC-Bus (IIC/I2C/I^2C) Anschluß. Diese Funktion wird von einem integrierten Hardwaremodul des Controllers unterstützt, eine Emulation durch Software erübrigt sich somit.

Soll das IIC-Businterface genutzt werden, sind an den beiden Bussignalen (SDA, SCL) Pull-Up Widerstände vorzusehen. Wenn diese nicht bereits außerhalb des Controllermoduls angesiedelt sind, können sie optional direkt auf dem S12compact bestückt werden (R10, R11).

Die IIC-Bussignale stehen an X4/65+66 zur Verfügung.

CAN-Interface

Der MC9S12DG256 verfügt über zwei CAN-Module, die mit CAN0 und CAN4 bezeichnet werden.

CAN0 kommuniziert über die Portpins PM0 und PM1 mit einem on-board CAN-Interface Chip (IC5), welcher das physische Businterface bildet. Die CAN-Bussignale CANH und CANL sind dann an X4/63 und X4/64 abzugreifen.

Wenn der S12compact der letzte Knoten am CAN-Bus ist, wird eine Terminierung erforderlich. Sie kann durch Schließen der Lötbrücke BR2 aktiviert werden.

Das Businterface für CAN4 ist nicht auf dem S12compact realisiert, es muß ggf. extern in der Anwenderschaltung realisiert werden. Die TTL-Signale für CAN4 werden über die Portpins PJ6 und PJ7 geleitet.

Hierbei ergibt sich eine Überschneidung mit der IIC-Funktion. Wenn beide Funktionen (IIC und CAN4) genutzt werden sollen, kann man CAN4 mittels MODRR auf PM4/5 oder PM6/7 umleiten, allerdings mit Einschränkungen der Nutzbarkeit des IF-Anschlusses bzw. der USB-Option.

Businterface

Das Businterface des HCS12 wird durch die Leitungen von Port A, Port B und Port K gebildet. Hinzu kommen einige Steuerleitungen des Port E. Diese Bussignale sind in der Schaltung des S12compact nicht belegt und stehen an X7 und X8 zur Verfügung.

Es handelt sich hierbei um SMD-Pads auf der Bestückungsseite der Platine, welche die Bestückung von doppelreihigen SMD-Stiftleisten im 2mm-Raster ermöglichen. Die beiden Steckverbinder sind dann in der Lage, einer aufgesetzten Huckepackplatine den erforderlichen mechanischen Halt zu bieten.

Es ist geplant, für den S12compact eine RAM-Platine anzubieten, die (zusammen mit einem BDM-Debugger) die Entwicklungsarbeit mit einem speziellen Flash-Emulationsmode unterstützt.

Verhalten nach Reset

Sobald die Resetleitung des Controllers freigegeben wird, holt sich die MCU die Information, an welcher Adresse das Programm des Anwenders beginnt. Der Controller liest hierzu den Resetvektor von den Speicherzellen $FFFE und $FFFF und springt dann an die dort angegebene Programmadresse.

Im Auslieferungszustand des S12compact ist im Flash-Bootblock ($E000-$FFFF) das Monitorprogramm D-Bug12 abgelegt. Der Resetvektor verweist auf den Beginn dieses Monitorprogramms. In Folge dessen startet nach jedem Reset automatisch D-Bug12.

Die Funktionen des D-Bug12 Monitors von Motorola sind in einem separaten Manual dokumentiert (siehe Produkt-CD).

Startup-Code

Jede Controllerfirmware beginnt mit einer Reihe von Anweisungen zur Initialisierung der Hardware. Im Fall des S12compact beschränken sich die *unbedingt notwendigen* Initialisierungen auf das Setzen des Stackpointers.

Die Abschaltung (bzw. ggf. die geeignete Initialisierung) des Watchdogs war bei früheren HC12-Derivaten zwingend notwendig. Beim MC9S12DG256 hingegen ist der Watchdog nach Reset zunächst stets disabled.

Zusatzinformationen im Web

Wenn zusätzliche Informationen zu Hard- und Software des S12compact vorliegen, veröffentlichen wir diese auf unserer Website:

http://www.elektronikladen.de/s12compact.html

7. Memory Map

Die Memory Map des MC9S12DG256 kann vom Anwenderprogramm zur Laufzeit modifiziert werden. Unmittelbar nach Reset weist die MCU folgende Speicherbelegung auf:

Start	Ende	Belegung
$0000	$03FF	Steuerregister
$0400	$0FFF	4KB EEPROM (Bereich $0000-$03FF = 1KB verdeckt durch Steuerregister)
$1000	$3FFF	12KB RAM
$4000	$7FFF	16KB Flash (entspricht Page $3E)
$8000	$BFFF	16KB Flash (beliebige Page $30..$3F, steuerbar über PPAGE)
$C000	$FFFF	16KB Flash (entspricht Page $3F)

Anhang

S-Record Format

Das von Motorola publizierte S-Record Format ist ein Dateiformat zur Definition von Objektdateien (Maschinencode, Executables) unter Verwendung einer textuellen (ASCII-) Notation, die es erlaubt, diese Objektdateien mit jedem beliebigen Texteditor zu betrachten oder zu ändern. Eine S-Record Datei besteht aus einer beliebigen Anzahl S-Records bzw. Zeilen. Eine jede Zeile hat die folgende logische Struktur:

ID	LEN	ADDR	DATA	CS	<EOL>

Das Feld ID gibt den S-Record Typ an. Relevant sind die Typen "S1", "S9" und gelegentlich "S0" (Kommentarrecord). Außer dem ID Feld bestehen alle weiteren Felder aus Paaren von Hexziffern, beispielsweise "A9", "55" oder "0F".

Das Feld LEN besteht aus einem derartigen Paar und bestimmt die Anzahl der folgenden Ziffernpaare (enthält die Ziffernpaare der Felder ADDR, DATA und CS).

ADDR ist die Anfangsadresse der Datenbytes dieser Zeile. Das Feld besteht aus zwei Byte (erst H-, dann L-Byte), d.h. aus zwei Ziffernpaaren.

DATA enthält die eigentlichen Codebytes, die das Maschinenprogramm bilden. DATA umfaßt (LEN - 3) Bytes bzw. Zeichenpaare.

Im Feld CS ist eine Prüfsumme enthalten. Sie wird gebildet aus den Werten der Zeichenpaare der Felder LEN, ADDR und DATA. CS ist das (niederwertigste Byte des) Einerkomplement der Summe aller vorgenannten Werte. EOL schließlich steht symbolisch für den durch CR, LF ($0D, $0A) gebildeten Zeilenvorschub.

Ein Beispiel soll die Handhabung verdeutlichen:

S1	13	2000	13A400262741010167CC10FF05C7A501	D1

Dieser S1-Record definiert $13-3 = $10 Bytes ab Adresse $2000 des Zielsystems. Die Ziffernpaare des DATA Feldes ergeben eine Summe von $04FB. Addiert man die $13 aus dem LEN Feld sowie $20 und $00 aus dem ADDR Feld hinzu, ergibt sich ein Wert von $052E. Das Einerkomplement des LSB ($2E) ergibt $D1. Dies ist der korrekte Wert für das Prüfsummenfeld.

Neben den S1-Records, welche die eigentlichen Daten enthalten, wird auch der S9-Typ verwendet. Dieser Typ beendet eine Serie von S1-Records. Abgesehen von dieser Terminierungs-Funktion kann in einem S9-Record die Startadresse des Programms vermerkt werden. Der Aufbau des S9-Records entspricht dem S1 Typ, wobei jedoch das Feld DAT leer bleibt. Das Feld ADDR spezifiziert die Startadresse des Programms. Ist hier $0000 eingetragen, wird angenommen, daß die Adresse des ersten geladenen Codebyte gleichzeitig die Startadresse des Programms ist. Ein typischer S9-Record sieht wie folgt aus:

S9	03	B600	46

EMV Hinweise

Die Baugruppe entspricht den EMV-Vorschriften. Zur Stromversorgung ist sie an einer Batteriespannungsquelle mit 5,0 Volt (Einhaltung der Spannungsgrenzwerte beachten!) oder an ein Netzteil mit CE-Kennzeichnung anzuschließen. Der Einsatz einer Mikrocontrollerplatine geht stets einher mit einer mehr oder minder umfangreichen Modifikation der Baugruppe (spezielle Firmware, angeschlossene Peripheriebauteile). Der Hersteller kann den vom Kunden geplanten Einsatz der Baugruppe nicht vorhersehen und daher auch keine Vorhersagen über die EMV-Eigenschaften der modifizierten Baugruppe machen. Anwender ohne Zugriff auf ein EMV-Prüflabor sollten die folgenden Richtlinien beachten, die in der Regel eine einwandfreie Funktion der modifizierten Baugruppe gewährleisten:

Um sicherzustellen, daß die Baugruppe auch dann den EMV-Vorschriften entspricht, wenn Verbindungsleitungen zu anderen Geräten (z.B. Personalcomputer) angeschlossen werden oder die Baugruppe vom Kunden selbst mit weiteren Bauteilen nachgerüstet wird (z.B. Meßadapter oder Leistungsendstufen), empfehlen wir, die komplette Baugruppe in ein allseitig geschlossenes Metallgehäuse einzusetzen.

Wird ein LC-Display angeschlossen (ebenfalls auf CE-Kennzeichnung achten), so darf das Verbindungskabel nicht länger als 10 cm sein; hier ist auf jeden Fall ein Metallgehäuse vorzusehen. Wenn für die Programmentwicklung oder die spätere Anwendung die RS232 Schnittstelle benötigt wird, so ist ein max. 10cm langes Kabel zur Verbindung mit der Anschlußbuchse zu verwenden. Die geschirmte Anschlußbuchse ist fest mit dem Metallgehäuse zu verschrauben. Extern zur Verbindung verwendete Anschlußkabel müssen, ebenso wie der Hostrechner (PC), mit dem CE-Zertifizierungszeichen versehen sein.

Es wird darauf hingewiesen, daß der Anwender selbst dafür verantwortlich ist, daß eine veränderte, erweiterte, mit anderen als vom Hersteller gelieferten IC´s bestückte oder mit Anschlußkabeln versehene Baugruppe den EMV-Vorschriften entspricht.